8-2011

Compact Co-location Pattern Mining

Mark Bow
Indiana University - Purdue University Fort Wayne

Graduate School ETD Form 9
(Revised 12/07)

PURDUE UNIVERSITY
GRADUATE SCHOOL
Thesis/Dissertation Acceptance

This is to certify that the thesis/dissertation prepared

By Mark Bow

Entitled
Compact Co-location Pattern Mining

For the degree of Master of Science

Is approved by the final examining committee:

Jin Soung Yoo
Chair

David Liu

Peter Ng

To the best of my knowledge and as understood by the student in the *Research Integrity and Copyright Disclaimer (Graduate School Form 20)*, this thesis/dissertation adheres to the provisions of Purdue University's "Policy on Integrity in Research" and the use of copyrighted material.

Approved by Major Professor(s): Jin Soung Yoo

Approved by: David Liu 07/18/2011
Head of the Graduate Program Date

Graduate School Form 20
(Revised 9/10)

PURDUE UNIVERSITY
GRADUATE SCHOOL

Research Integrity and Copyright Disclaimer

Title of Thesis/Dissertation:

Compact Co-location Pattern Mining

For the degree of Master of Science

I certify that in the preparation of this thesis, I have observed the provisions of *Purdue University Executive Memorandum No. C-22,* September 6, 1991, *Policy on Integrity in Research.**

Further, I certify that this work is free of plagiarism and all materials appearing in this thesis/dissertation have been properly quoted and attributed.

I certify that all copyrighted material incorporated into this thesis/dissertation is in compliance with the United States' copyright law and that I have received written permission from the copyright owners for my use of their work, which is beyond the scope of the law. I agree to indemnify and save harmless Purdue University from any and all claims that may be asserted or that may arise from any copyright violation.

Mark Bow

Printed Name and Signature of Candidate

07/18/2011

Date (month/day/year)

*Located at http://www.purdue.edu/policies/pages/teach_res_outreach/c_22.html

COMPACT CO-LOCATION PATTERN MINING

A Thesis

Submitted to the Faculty

of

Purdue University

by

Mark Bow

In Partial Fulfillment of the

Requirements for the Degree

of

Master of Science

August 2011

Purdue University

Fort Wayne, Indiana

TABLE OF CONTENTS

LIST OF FIGURES

ABSTRACT

Bow, Mark. M.S., Purdue University, August 2011. Compact Co-location Pattern Mining. Major Professor: Jin Soung Yoo.

With the advent of data gathering and analysis, many different domains such as public health, business, transportation, geology and so on are generating large volumes of data. Such large sets of data could contain potentially interesting patterns that can provide useful information to these domains. Many techniques in the area of data mining have been employed to discover useful patterns. Spatial co-location pattern mining has been popularly studied in spatial data mining area.

Spatial co-location patterns represent the subset of spatial events whose instances are frequently located together in nearby geographic space. A common framework for mining spatial co-location patterns employs a level-wised Apriori-like search method to discover co-locations and generates redundant information by searching all 2^l subsets of each length l event set. As the number of event types increases, the search space exponentially increases. This adversely increases the computational time of mining for co-location patterns. In this thesis, we propose two problems for mining compact co-locations which concisely represent co-location patterns. The first problem addresses mining for maximal co-located event sets. The second problem addresses top-k closed mining which finds k closed co-located event sets which have higher prevalence values than other closed co-locations. This thesis develops an algorithm for each problem to efficiently search for the compact patterns. The experiment results show that our algorithms are computationally efficient in finding the compact co-location patterns.

1 INTRODUCTION

Areas such as mobile computing, scientific simulations, business science, environmental observation, climate measurements, geographic search logs, and so on are producing large, rich spatial data sets. Manual analysis of these large data sets becomes impractical and there is a need for computational analysis techniques for automatic extraction of potential valuable information. Spatial data mining [1, 2] has been employed to discover interesting and previously unknown, but potentially useful patterns from large spatial data sets.

Spatial co-location pattern mining, as one important area in spatial data mining, has been prominently researched in data mining literature [3–10]. Spatial co-location pattern describes "a set of spatial events which are frequently observed together in a spatial proximity" [3]. An example of co-location patterns can be found in the following areas:

- Disease control and public health: The pattern of "poor mosquito control site and the present of birds implies human case of West Nile disease" can be formulated based on frequency of geographic cases of West Nile virus and poor mosquito control site with the presents of birds.

- Transportation: Based on some spatial properties, patterns can be inferred of frequent points of traffic accidents, traffic jams, and police zones.

- Business: A planner who is responsible for the arrangement of a new location of a business store would like to know the different types of neighboring spatial objects frequently appearing together. With this knowledge, the planner can determine the profitability of similar stores and its surrounding objects to make a better decision relating to the localization of a new store.

- Social Science: Ecologists may be interested in the co-occurs of environmental or social factors that affect animal behaviors.
- Geology: Geologists are interested in the relationship of spatial distribution among various kinds of minerals elements.
- Mobile Computing: A mobile company may be interested in knowledge of finding requested services ordered by users located close to one another in the neighboring area. This knowledge of frequently requested services together in a nearby area can be used for providing location based services. It can also be used for providing attractive location-sensitive advertisements, recommendations, and so on.

Figure 1.1 shows an example of co-location patterns in mobile application domain. Say that a mobile service provider is interested in knowing what services are requested frequently in nearby neighborhood area. In Figure 1.1 (a), there are a few different services in the area, i.e. movie, restaurant, sales, weather, and news. These requests are recored in the service provider's log database like Figure 1.1 (b). In each log, entries describe the position of where the service was requested, the type of service requested, the time that the service was requested and so on. Using the data in the log database, the service provider can determine patterns in spatial proximity. Figure 1.1 (c) shows based on some frequency, that the pattern {Sales, Restaurant} and {Movie, Restaurant} are spatial co-located events sets that frequently located in the nearby area.

Figure 1.1. An Example of Co-location Patterns in Mobile Application Domain

1.1 Challenges

Spatial co-location mining is a more extensive problem of mining association patterns. In classical data mining literature, there are many works for discovering frequent patterns from transactional datasets [11]. However, it is non-trivial to re-use them directly in a co-location context because, unlike market-basket data, transactions are not explicit in spatial data. Spatial data contains properties and relationships that are distinct from transactional data. For example, spatial data are embedded in continuous space rather than the discrete space of numeric and categorical data typically found in transactional data. Also spatial data share a variety of spatial relationships among each other as contrast to transactional data in which transactions are independent of each other. Extracting spatial patterns from spatial data is more difficult than extracting the corresponding patterns from traditional data because of the complexity of data type and implicit relationships among spatial objects.

State-of-the-art algorithms for spatial co-location pattern mining [3–5, 7, 8] use a generation-and-test method to discover the interesting patterns. Such algorithms generate redundant co-location patterns by searching all 2^l subsets of each length l event set since they are too co-location patterns. As the number of event types increases,

the search space exponentially increases. This exponential complexity increases the computational time of mining for co-location patterns and can restrict the algorithm to discovering only short patterns.

In lieu of this, there has been interest in mining only compact result set of co-location patterns. In this thesis, we study two new problems in co-location pattern mining, namely, maximal co-location mining and top-k closed co-location mining. A co-located event set is maximal if and only if it is prevalent and it does not have any prevalent supersets. A co-location event set is closed if and only if it has no superset with the same prevalence value. Top-k closed co-locations are k closed co-locations with higher prevalence values than other closed co-locations. The result set of mining only top-k closed or maximal event sets is a much smaller than the result set that enumerates all possible co-location subsets.

1.2 Thesis Contribution

- We propose two new co-location patterns. One is a maximal co-location pattern and the other is a top-k closed co-location pattern.
- We present a common framework for mining these co-location patterns.
- We develop for each co-location pattern an efficient algorithm.
- We conduct experiments using our algorithms on real datasets. The experimental results shows that our algorithms are computationally more efficient than a general co-location mining algorithm for finding the co-location patterns.

The remainder of this thesis is organized as follows. Chapter 2 describes the basic concept of spatial co-location mining. Chapter 3 defines two new co-location patterns and gives the problem statement to mine each pattern. It also includes the related works. Chapter 4 describes a common framework for finding the compact co-location patterns. Chapter 5 presents an algorithm of maximal co-location pattern mining and

the experimental result. Chapter 6 presents an algorithm of top-k closed co-location pattern mining and the experimental result. Chapter 7 ends with the conclusion.

2 BASIC CONCEPT OF SPATIAL CO-LOCATION MINING

This chapter describes the basic concepts of spatial co-location mining.

Definition 2.0.1 Spatial Event: *A feature or event of interest.*

An example of spatial events would be any points of interest such as schools, churches, sub-divisions, and airports. This thesis is concerned with how these spatial event frequently occur together and how they related to each other on space.

Definition 2.0.2 Spatial Co-location: *Given spatial event types, E and their corresponding instance objects S and a neighboring relationship R over S, a co-location X (or co-located event set X) is the subset of spatial events, $X \subseteq E$, whose instance objects frequently form cliques under the neighboring relationship R.*

Figure 2.1 shows an example spatial data set. There are three different event types $E = \{A, B, C\}$ each with object instances denoted by the event type and a numeric id value e.g. A.1. Line edges are shown among the objects indicting neighboring

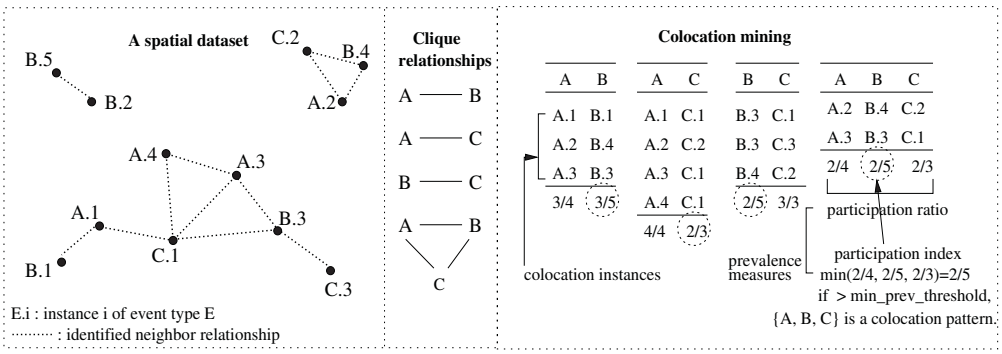


Figure 2.1. Spatial Co-location Pattern

relationships if they meet a given distance threshold, for example 0.2 miles. Given three event types, there are four possible event sets, {A, B}, {A, C}, {B, C} and {A, B, C}.

Definition 2.0.3 Co-location instance: *A co-location instance I of a co-location C is the set of objects $I \subseteq S$, which includes all event types in the co-location and forms a clique.*

For instance, in Figure 2.1, {A.1, B.1}, {A.2, B.4}, and {A.3, B.3} are co-location instances of {A, B}. The prevalence strength of a co-location instance is often given by its participation index [3].

Definition 2.0.4 Participation Index: *The participation index $PI(X)$ of co-location X is defined as $PI(X) = min_{e_i \in X}\{Pr(X, e_i)\}$*

Definition 2.0.5 Participation Ratio: *The participation ratio of event type e_i in a co-location $X = \{e_i, ..., e_k\}$ is the fraction of objects of event e_i in the neighborhood of instances of co-location $X - e_i$, i.e., $Pr(X, e_i) = \frac{Number\ of\ distinct\ objects\ of\ e_i\ in\ instances\ of\ X}{Number\ of\ objects\ of\ e_i}$*

For example, in Figure 2.1, event type A has four co-location instances, event type B has five co-location instances, and event type C has three co-location instances in the spatial data set. The co-location X ={A, B, C} has instances of {A.2, B.4, C.2} and {A.3, B.3, C.1}. The participation ratios of co-location X can be calculated from the co-location instances. Event type A has a participation ratio $PR(X, A)$ of $\frac{2}{4}$ since only A.2 and A.3 are among the four instances that participate in the co-location X. $PR(X, B)$ and $PR(X, C)$ is $\frac{2}{5}$ and $\frac{2}{3}$ respectively. The participation index of X is the minimum of the ratios for which $PI(X) = \frac{2}{5}$. Now, depending on the what value the minimum threshold is, say $\frac{1}{5}$, the co-location X is considered frequent if $PI(X)$ is greater than the threshold.

3 PROBLEM STATEMENT AND RELATED WORK

This chapter presents the problem statements of mining maximal co-location patterns and top-k closed co-location patterns. This chapter ends with related work.

3.1 Problem Statements

We first define maximal co-location mining problem:

Definition 3.1.1 Maximal co-location event set *is an event set, X, that is prevalent and has no prevalent supersets. X is prevalent when X 's participation index is greater than the user-define minimum threshold.*

Using Figure 2.1 as an example again, lets says that the minimum prevalence threshold is $\frac{1}{5}$. Examining the co-locations, we see that {A, B, C} has a prevalence participation index of $\frac{2}{5}$ which is above the minimum prevalence threshold and thus is prevalent. Also we see that the co-location {A, B, C} has no supersets that are prevalent. This makes this event set a maximal co-location. Note that {A, B}, {A, C}, and {B, C} are prevalent but are subsets of co-location {A, B, C}. Therefore, they are not maximal co-locations.

The problem statement for maximal co-location pattern discovery is as follows:

Given

1) A set of spatial events $E = \{e_1, ..., e_m\}$

2) A dataset of spatial point objects $S = S_1 \cup ... \cup S_m$ where $S_i (1 \leq i \leq m)$ is a set of objects of event type e_i. Each object $o \in S_i$ has a vector information of {event type e_i, object id j, location(x, y)}, where $1 \leq j \leq |S_i|$.

3) A spatial neighbor relationship.

4) A minimum prevalent threshold θ

Develop

An algorithm to find maximal co-location patterns efficiently in computation.

The following is the problem statement of top-k closed co-location pattern mining:

Definition 3.1.2 Closed co-locations *is a event set, X, that is closed if there exists no proper superset $X\prime \supset X$ such that $PI(X\prime) = PI(X)$.*

Here is a simple example of closed co-location: Assuming that {A, D} is frequent with a participation index of $\frac{1}{2}$. Its superset, {A, C, D} is frequent and also has a participation index of $\frac{1}{2}$. Then by definition 3.1.2, {A, D} is not a closed co-location. Now the co-location {A, C, D} could be a closed co-location if it satisfies definition 3.1.2.

Definition 3.1.3 top-k closed co-locations: *Let L be a list of all closed co-locations by descending their participation index values, and let p be the participation index of the kth closed co-location in the list L. The top-k closed co-locations are a set of closed co-locations having participation index $\geq p$.*

For example, suppose k is 2. In Figure 2.1, top-k closed co-locations are {A, B}, {A, C} since they are higher prevalence values than other closed sets.

The problem statement for top-k closed co-location mining is as follows:

Given

1) A set of spatial events $E = \{e_1, ..., e_m\}$

2) A dataset of spatial point objects $S = S_1 \cup ... \cup S_m$ where $S_i (1 \leq i \leq m)$ is a set of objects of event type e_i. Each object $o \in S_i$ has a vector information of {event type e_i, object id j, location(x, y)}, where $1 \leq j \leq |S_i|$.

3) A spatial neighbor relationship.

4) The number of closed co-location to mine for, k.

Develop

An algorithm to find top-k closed co-locations efficiently in computation.

3.2 Related Works

The related works are divided into the following category: spatial data mining, spatial co-location pattern mining, maximal pattern mining, and top-k closed pattern mining.

3.2.1 Spatial Data Mining

Spatial association rule mining problem was first discussed in [8]. The paper discovers the subsets of spatial features frequently associated with a user specific reference feature. Castro et al. [12] proposed a clustering-based map overlay approach for discovering spatial association patterns.

3.2.2 Spatial Co-location Mining

Shekar et al. [3] formulated the co-location pattern mining problem and developed a co-location mining algorithm with join operations that that are used to find co-location instances. Subsequent works improved the join operations were proposed in [4] and [13]. Xiao et al. [10] proposed a density based approach to searching for co-location instances. Zhang et al. [14] enhanced searching for co-location patterns proposed in [3] by presented an approach to find spatial star, clique, and generic patterns. Morimoto [7] discovers frequent patterns in spatial database by grouping neighborhood class sets using a support count measure as a means to determine frequency. From our previous works [15, 16], we proposed a problem to find N-most prevalent co-located event sets per each pattern.

3.2.3 Maximal Pattern Mining

In general data mining literature, MaxMinter [17] algorithm discovers maximal item sets. The algorithm scales roughly linearly in the number of maximal pattern embedded in a database regardless of the length of the longest patterns. Zaki et al. [18] propose MaxEclat and MaxClique algorithms for identifying maximal item sets. They attempt to divide the subset lattice into smaller pieces and apply a bottom-up Apriori traversal with a vertical data representation. The Pincher-Search algorithm [19], proposed by Lin et al., mines for long maximal item sets. The algorithm prunes the search space both from the top and bottom. MAFIA [20] is a recent algorithm that searches for maximal patterns. The algorithm makes use of three pruning strategies and a vertical bit-vector data format and applies optimizations of compression and projection of bitmaps to improve performance.

3.2.4 Top-k Pattern Mining

In general data mining literature, Pietracaprina et al. [21] proposed TopKMiner algorithm, an incremental approach to finding top-k closed item sets without a minimum threshold. A unique feature of the algorithm allows users to dynamically increase the number of item sets by increasing k without restarting the algorithm. Songram et al. [22] presents TOPK_CLOSED algorithm that mines top-k closed item sets. The algorithm uses a best-first search that examines the highest support values first and stops until it comes across an item set have a lower support k^{th}. Wang et al. [23] proposed TFP algorithm that mines top-k closed item sets using a user-specified k and user-specified minimum length l of the item sets. The algorithm does not require the user to specify a minimum threshold but keeps an internal threshold and increments it during the mining process.

4 A FRAMEWORK OF MINING COMPACT CO-LOCATIONS

The data mining process is first depicted in Figure 4.1. Each box represents a phase in the mining process and each arrow represents the transition of one phase feeding into the next phase and then repeating back to the beginning of the process. The phases are as follows:

1. Determine Domain: The beginning phase of mining process is the determination of what application domain to target. This phase also includes addressing the data mining problem(s), what data to gather,which mining algorithm to use, and initial hypotheses.

2. Gather Data: This phase gathers the actual data to be processed and assesses the characteristics and quality of data. This phase also determines what mining parameters to set before running the algorithm.

3. Run Algorithm: Depending what algorithm was chosen, this phase runs the algorithm finding interesting patterns.

4. Validate Output: Review the generated patterns from the result set with data domain experts.

This process repeats as many times as necessary, going back to the data gathering phase. Figure 4.2 shows our framework of mining compact co-location patterns.

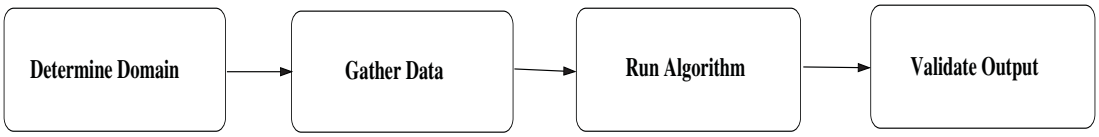


Figure 4.1. Data Mining Process

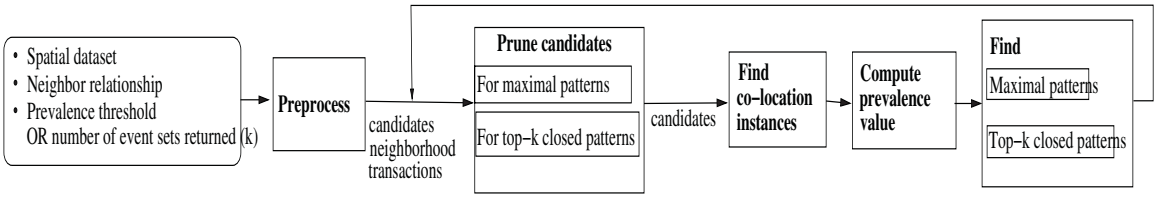


Figure 4.2. A Framework of Compact Co-location Pattern Mining

1. Preprocessing: This phase takes raw spatial data and transforms it into neighboring transactions so that the mining algorithm can easily process.

2. Candidate Generation: Based on the transaction dataset from the previous step, generate co-location candidates and prune out any candidate co-locations that do not form a clique relationship.

3. Maximal Mining: Process any candidates from the previous phase and find all maximal result sets.

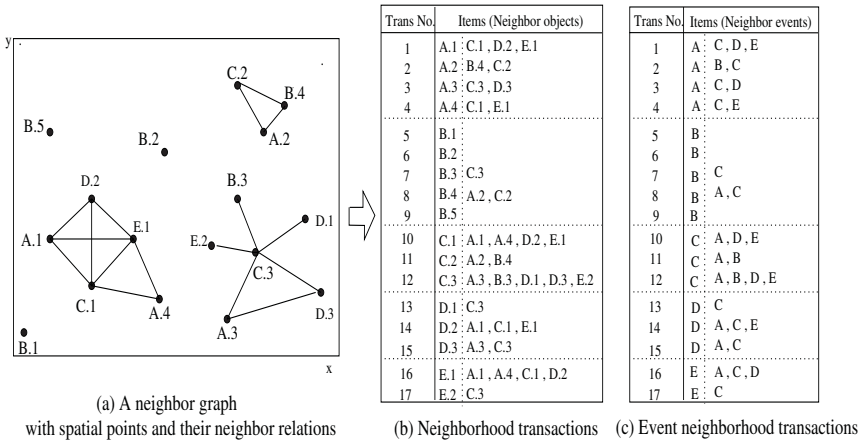
4. Top-k Closed Mining: Process any candidates from the previous phase and find top k closed result sets.

4.1 Preprocessing

The first phase in our framework for compact co-location mining is preprocessing input data.

One of the input that feeds into the preprocessing step is a spatial data set containing spatial objects and their connected edge to other objects that represents neighborhood relationships over a neighboring graph as depicted Figure 4.3 (a). We present the input spatial data as a set of neighborhood transactions.

Definition 4.1.1 *Given a spatial object $o_i \in S$, the* **neighborhood transaction** *of o_i is defined as a set of spatial objects $\{o_i, o_j \in S | R(o_i, o_j) = true \wedge o_i's$ event type $\neq o_j's$ event type$\}$, where R is a neighbor relationship.*



Trans No.	Items (Neighbor objects)
1	A.1 , C.1 , D.2 , E.1
2	A.2 , B.4 , C.2
3	A.3 , C.3 , D.3
4	A.4 , C.1 , E.1
5	B.1
6	B.2
7	B.3 , C.3
8	B.4 , A.2 , C.2
9	B.5
10	C.1 , A.1 , A.4 , D.2 , E.1
11	C.2 , A.2 , B.4
12	C.3 , A.3 , B.3 , D.1 , D.3 , E.2
13	D.1 , C.3
14	D.2 , A.1 , C.1 , E.1
15	D.3 , A.3 , C.3
16	E.1 , A.1 , A.4 , C.1 , D.2
17	E.2 , C.3

Trans No.	Items (Neighbor events)
1	A , C , D , E
2	A , B , C
3	A , C , D
4	A , C , E
5	B
6	B
7	B , C
8	B , A , C
9	B
10	C , A , D , E
11	C , A , B
12	C , A , B , D , E
13	D , C
14	D , A , C , E
15	D , A , C
16	E , A , C , D
17	E , C

Figure 4.3. Preprocess: Neighborhood Transaction

For example, in (b) of Figure 4.3 the neighborhood transaction of A.1 is {A.1, C.1, D.2, E.1}. Each object in the neighborhood transaction has a neighbor relationship to the first object, A.1. A.1 is considered a reference object. Using neighborhood transaction provides several advantages for mining co-location patterns. First, this representation does not lose any objects and neighboring relationships among objects in the input data. Second, the upper bound on participation index can be inferred from the representation of co-located event set. Finally, it can be used to filter candidate event sets. The set of distinct event types from a neighborhood transaction is called a **event neighborhood transaction**. Figure 4.3 (c) shows the event neighborhood transactions. Co-location candidates can be generated from the event neighboring transactions.

4.2 Candidate Generation

Before taking the effort of generating all candidates, it would be ideal to only generate candidates having at least one co-location instance. If we can detect which event sets form clique instances, we can reduce the number of candidate sets to exam, thus reducing the computational time searching the instances of candidates. For example

in Figure 4.3 (a), the event set {A, B, C} does not form a clique relationship. Knowing this, there is not a need to generate candidates for this event set. For those event sets that have clique relationships, co-location candidates can be generated from these event sets from the event neighboring transactions. The candidate generation scheme proposed in [16] is used for candidate generation phase of the mining process. The basic idea is to build an event tree per each event type from the event neighborhood transactions.

Definition 4.2.1 *A* **reference event pattern tree** *(or* **event tree** *in short) is a tree structure defined: (1) It consists of one root labeled as a reference event type, a set of event prefix subtrees as the children of the root, (2) Each node consists of three fields: event-type, count, and node-link, where event-type denotes a event this node represents, and count registers the number of neighborhood transactions represented by the portion of the path reaching this node. The transactions start with an object whose event type is the same with the event type of the root note. Node-link links to the next node in the tree carrying the same event-type.*

The reference event pattern tree is an variant of the FP-tree in [16]. FP-tree is a data structure that can store compressed information about frequent patterns without having to generate candidates. However, the FP-tree cannot be used directly in our mining process because of storing instances having clique relationships. Instead, the reference event pattern tree is used to store candidate information and to filter candidate event sets during the mining process.

Figure 4.5 shows the candidate generation process. Part (a) shows the event trees which are generated from the event neighborhood transactions. To get an idea of the event tree generation take a look at Figure 4.4. Figure 4.4 shows an event tree of event C. Using C as the root of the event tree, each neighborhood transaction row which starts with C is examined and the object nodes are generated from the root node. When there is a match for the same node, that node's support count is incremented. Applying the same event tree generation to the other neighborhood transaction groups forms the rest of the event tree in Figure 4.5 (a).

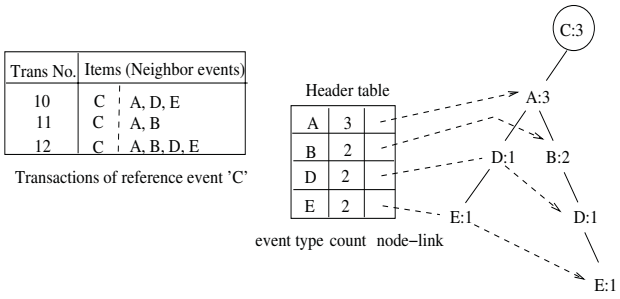


Figure 4.4. Event Tree Generation from Neighborhood Transactions

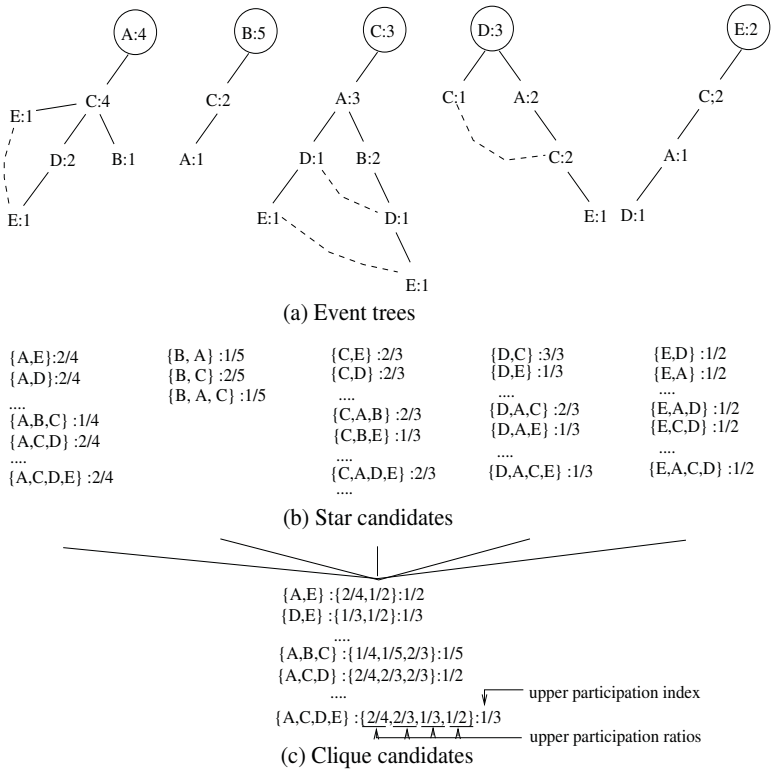


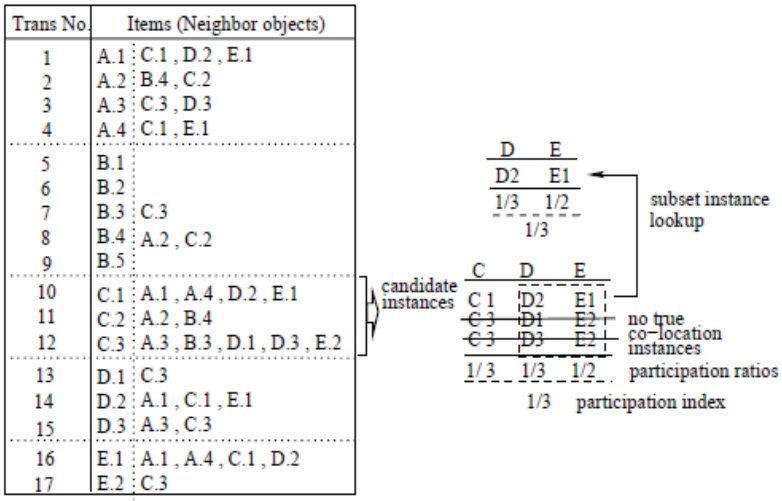
Figure 4.5. Candidate Generation

From the event trees, candidates can be generated using the FP-Growth algorithm [24]. Figure 4.5 (b) shows the event sets generated from each event tree. The event sets generated is called **star candidate** sets. The star candidates contains support count information of the event set. The support count serves a purpose of

determining how frequent the first event has a neighboring relationship with the other events. For example the event set of {B, A, C} has a support count of $\frac{1}{5}$, meaning that a occurrence of B having a single neighbor relationship with both A and C.

After generating star candidates sets, we combine them for filtering co-location candidates. For example, a set,{A, B, C} can be a co-location candidate, if each event in the set have neighbor relationships with the other events. To determine whether {A, B, C} is a co-location candidate, we would check for {A, B, C} in the set of A star candidates, {B, A, C} in the set of B star candidates, and {C, A, B} in the set of C star candidates. {A, B, C} becomes a co-location candidate from the results in Figure 4.5 (b). As another example, a set {B, C, E} is not a co-location candidate because there is no {B, C, E} in the set of B star candidates and no {E, B, C} in the set of E star candidates even if {C, B, E} is in the set of C star candidates. Figure 4.5 (c) shows the combined candidate set. A co-location candidate inherits the support count of each event from its star candidates.

For example, {A, B, C} has the frequency values of A, B, and C events, i.e., $\frac{1}{4}$, $\frac{1}{5}$ and $\frac{2}{3}$ from each star candidate {A, B, C}: $\frac{1}{4}$, {B, A, C}: $\frac{1}{5}$ and {C, A, B}: $\frac{2}{3}$. The minimum value amoung the three values, $\frac{1}{5}$, becomes the upper bound on participation of co-location {A, B, C}.



(a) neighborhood transactions (b) candidate instances and true instances

Figure 4.6. Co-location Instance Search

4.3 Instance Filtering

Once candidates are filtered for co-location patterns, the next steps gather their co-location instances. When these instances are found we can determine their true participation index. The neighborhood transactions from the preprocessing phase can be reused to find the co-location instances. The following term is defined for co-location instances.

Definition 4.3.1 *Let* $I = \{o_1, \ldots, o_k\} \subseteq S$ *be a set of spatial objects whose event types* $\{e_1, \ldots, e_k\}$ *are different each other. If all objects in* I *are neighbors to the first object* o_1*,* I *is called the* **star instance** *of co-location* $C = \{e_1, \ldots, e_k\}$.

Star instances of a candidate can be gathered from the neighborhood transactions with the first object's event type being the same as the first event type of the co-location. Figure 4.6 shows the star instances of {C, D, E} co-location being gathered from the neighborhood transactions in part (a). A true co-location instance can be filtered from the star candidates by examining the cliqueness of subset of the instance

not including the first object. For example, Figure 4.6 (b) shows that {C.1, D.2, E.1} is a true co-location instance since its subset, {D.2, E.1}, is a true co-location instance of {D, C}. In contrast, {C.3, D.1, E.2} and {C.3, D.3, E.2} are not true co-location instances since their subsets are not the co-location instances of {D, E}.

5 MAXIMAL CO-LOCATION PATTERN MINING

This chapter first describes the background and algorithmic design concepts of maximal co-location patterns. And then it presents maximal co-location pattern mining algorithm. The chapter ends describing the experiment results.

5.1 Background

Many spatial co-location mining algorithms are based on Apriori which uses a generate-and-test method [4, 5, 7, 8, 11]. These algorithms traverse the search space in a breadth-first manner searching for prevalent co-locations by enumerating co-location instances forming clique relationships. In order to generate co-locations of size l, all 2^l subsets are examined since they too must be included in the final result set. Increasing the number of event types effects the search space by increasing it exponentially. This increase in the search space causes computational performance problems since most of the computation time is used to find co-location instances forming clique relationships. When presented with long patterns in the dataset, it is often impractical to generate all of the co-locations. This computational complexity limits the current algorithms to small co-location pattern discovery.

Because of these problems, there has been interest in mining only maximal co-location patterns. A co-location is maximal if it satisfies definition 3.1.1. By mining only maximal patterns, the result set implicitly and concisely represents all co-location patterns.

5.2 Algorithmic Design Concept

Using a brute force approach to discover maximal co-location patterns would entail finding all co-location instances forming clique neighboring relationships, compute the participation index per event set, and return all maximal patterns. We would have to examine around 2^l event sets when l event types are given. As the number of l event types increases, the brute force approach quickly becomes impractical. Our algorithm focuses on reducing the number of candidate event sets examined for maximal co-location patterns. This algorithmic approach of reducing the search space eventually reduces the expensive operation to find co-location instances in the datasets. This section describes the algorithmic design concept for maximal co-location patterns.

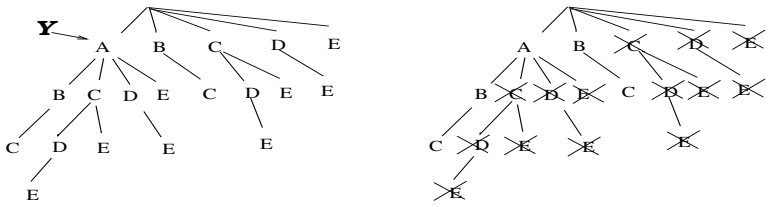
5.2.1 Preprocess and Candidate Generation

The preprocess and candidate generation steps were explained in depth in Chapter 3. The spatial data set are converted to a set of neighboring transactions and event neighboring transactions. From the event neighborhood transactions, co-located candidates are generated

5.2.2 Candidate Pruning

Figure 5.1 (a) shows the pattern search space with candidates generated from the candidate generation step using a lexicographic subset tree. In the subset tree, size l event set are ordered lexicographically on each level and all children are associated with the earliest subset in the previous level. The event set identifying each node will be refers to as the node's *head* , while possible extensions of the node are called the *tail*. For example, consider node **Y** in Figure 5.1. **Y**'s head is {A} and the tail is the set {B, C, D, E}. The head union tail (HUT) is {A, B, C, D, E}.

In addition to the filtering of co-location candidates represented so far, maximal co-location patterns have additional pruning schemes to further reduce the co-location



(a) A subset tree with clique candidates

(b) After finding a {A C D E} maximal set

Figure 5.1. Subset Pruning by Superset

candidates. The algorithm take a two part strategy, a depth-first and breath-first search of the search space. The first part is to do a depth first search of the search space to quickly find long maximal patterns. The algorithm starts with the longest l_{max} co-location candidates in the search space to determines if they are maximal as specified in definition 3.1.1. It is now that the algorithm can determine whether the candidates satisfy definition 3.1.1. Once the algorithm determines a candidate is a maximal, the algorithm can begin the pruning process. The second part involves a subset pruning scheme to reduce the search space futher.

With traversing the tree in breadth-first manner, the algorithm checks whether the HUT of each node is a subset of a current maximal set. If the HUT is a subset of any maximal candidates in the result set, the sub-tree whose root is the node is pruned. For example, Figure 5.1 (a), lets assume that {A, C, D, E} is a maximal candidate. Then the algorithm checks breadth-first for candidates to prune. In the first level, the HUT of a node A is {A, B, C, D, E}. Since it is not a subset of {A, C, D, E}, the algorithm cannot prune the sub-tree whose root is A. Next node in the level is node B which the HUT of node B is {B, C}. Since this candidate is a subset of {A, C, D, E}, the algorithm prunes that sub-tree. The sub-tree of Node C is also pruned. The pruning continues on with the rest of the nodes in the first level. Then

the algorithm starts pruning the second level where the HUT of node C is {C, D, E} is pruned. The algorithm keeps pruning away at the subset tree by supersets until the subset tree resembles Figure 5.1 (b).

5.2.3 Co-location Instance Search

Once the co-location candidates are generated and filtered out any irreverent candidates, we must find the co-location instances of each candidate and compute their true participation index. Reusing the neighborhood transaction from the previous step, we can gather co-location instances from star instance of the neighborhood transaction and filter those instances having an additional relationship within the subset of that instance. The true participation index of a co-location is calculated from the participation ratios of their instances. This mining step has been discussed at length in chapter 3.

5.2.4 Maximal Set Search

After calculating the true participation index of a given candidate from their co-location instances, we can then determine if the candidate is maximal and include it as the result set of maximal patterns. This determination is made by comparing the true participation index of a potential maximal candidate co-location to a user-defined prevalent threshold, θ, specified at the beginning of the mining algorithm. If the candidate's true participation index is greater than the threshold, θ, then the candidate is maximal and is inserted into the result set. Once a maximal pattern is found the pruning scheme is executed to prune out any subsets of that maximal pattern. Finding maximal patterns and running the pruning scheme happens at each l level until there are no more candidate to process.

5.3 Algorithm

This section shows the pseudo-code algorithm to mine maximal co-location patterns based on the algorithmic design concepts. The pseudo-code is shown in Algorithm 1.

Preprocess(Step 1): Given an input spatial dataset and a neighbor relationship, first find all neighboring object pairs using a geometric method such as plane sweep [25], or a spatial query method using quaternary trees or R-Trees [26]. The neighborhood transactions are generated by grouping the neighboring objects per each object.

Candidate Generation (Step 2-6): An CP-tree per each even type is constructed from the event neighborhood transactions of the reference event. Star candidates are generated using a project based mining algorithm (FP-growth) [27]. Co-location candidates are generated with combining the star candidates. The upper bound on participation index of a candidate is set to the minimum value of support values of events in the set.

Select length l (from l_{max} to 2) co-location candidates (Step 7-10): First, the longest length of candidates events is set to l_{max}. The maximal pattern mining is processed from length $l = l_{max}$. Select length l candidates from the candidate pool.

Gather the star instances of candidates (Step 11): The star instances of candidates are gathered from the neighborhood transactions whose first object's event type is the same with the first event of the candidate. The upper bound on participation index of the candidate is updated with its star instances. If the updated upper bound is less than the prevalence threshold, the candidate is no longer considered.

Filter the co-location instances of a candidates and compute the true participation index (Step 12-16): Next filter the true co-location instances from the star instances of a candidate examining all neighbor relationships of objects in a star instance except the first object. Here the algorithm uses the neighbor pair information generated in the preprocessing steps without geographic operations. After finding all true instances of a candidate, compute its true participation index.

Update the result set and prune the subsets (Step 17-21): If the candidate set's true participation index is greater than a prevalence threshold, insert the current candidate set into the result set, and all subsets of the maximal set are pruned.

Return the final result set (Step 22-23,24): The procedure of step 9 to step 23 is repeated until l reaches to 2 or there is no candidates left. Finally, return the final result set of maximal co-location patterns.

5.4 Experimental Evaluation

To validate our proposed approach to finding maximal co-location patterns, we ran an experimental evaluation to test for computation effectiveness. This section shows our maximal algorithm evaluated with that of a general co-location mining algorithm.

5.4.1 Experimental Setup

We named our maximal co-location algorithm as 'MaxColoc'. The general co-location algorithm is named 'GeneralColoc' which is found in [3]. We named the general co-location algorithm with a maximal post processing module 'GeneralColoc+ Maximal' and use it to compare 'MaxColoc' in our experiments.

We used the datasets containing points of interest in California [28] for our experiments. The number of original data points was 104,770 and the number of distinct event types was 40. We test 'MaxColoc' and 'GeneralColoc+Maximal' for the following experiments:

- *Experiment 1:* Number of candidates: We observe the number of candidates examined for co-location instances for each pattern length. For this experiment, we prepared dataset #1 which contains 40 event types, 12000 data points and a neighboring distance of 1000. The minimum prevalent threshold used is 0.1 for 'MaxColoc' and 'GeneralColoc+Maximal'.

- *Experiment 2:* By prevalent threshold: We examine the effects of runtime efficiency on different minimum prevalent threshold values of both algorithms. For this experiment, we prepared dataset #2 that contains 40 event types, 12000 points and a neighboring distance of 1000. We set the minimum prevalent threshold to 0.15, 0.20, 0.25, and 0.30.

- *Experiment 3:* By neighboring distance: We see the effects of runtime efficiency as neighboring distance increases. For this experiment, we prepared datasets #3, #4, #5, and #6 each with a neighboring distance of 800, 1000, 1400, 2000 respectively. These datasets all have 40 event type and 1200 points. We will test the experiment with a prevalent threshold of 0.1.

- *Experiment 4:* By the number of data points: We will see the effects of runtime efficiency as the number of data points increases. For this experiment, we prepared the datasets #7, #8, #9, #10, and #11 each having 3000, 6000, 12000, 24000, and 48000 respectively. Each experiment uses prevalent threshold of 0.1. All datasets have 40 event types and a neighboring distance of 1000.

All experiments were conducted on a Linux system with 2.0 GB main memory and the implementations of 'MaxColoc' and 'GeneralColoc+Maximal' are in C++.

5.4.2 Experimental Results

Experiment 1: In Figure 5.2, we see that the number of candidates to examine for co-location instances of 'MaxColoc' is less than the number of candidates of 'GeneralColoc+Maximal' for each pattern length starting from length of 2. As the the pattern length increases, we see that the number of candidates to examine decreases for both algorithms but the number of candidates for 'MaxColoc' is always less than that of 'GeneralColoc+Maximal'.

Experiment 2: As shown in Figure 5.3, as the prevalent threshold increases, the execution time decreases. Comparing 'MaxColoc' with 'GeneralColoc+Maximal' shows

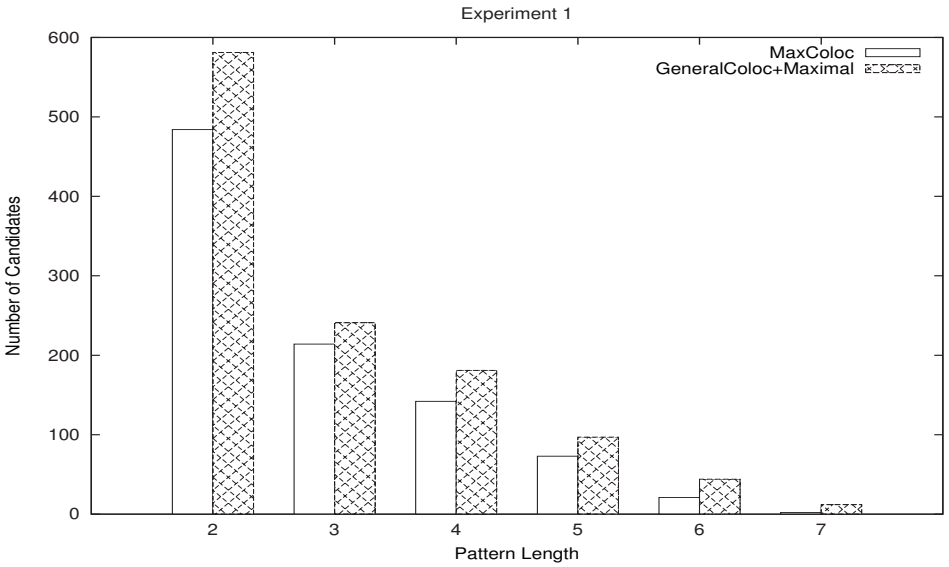


Figure 5.2. Experiment 1: Number of Candidates

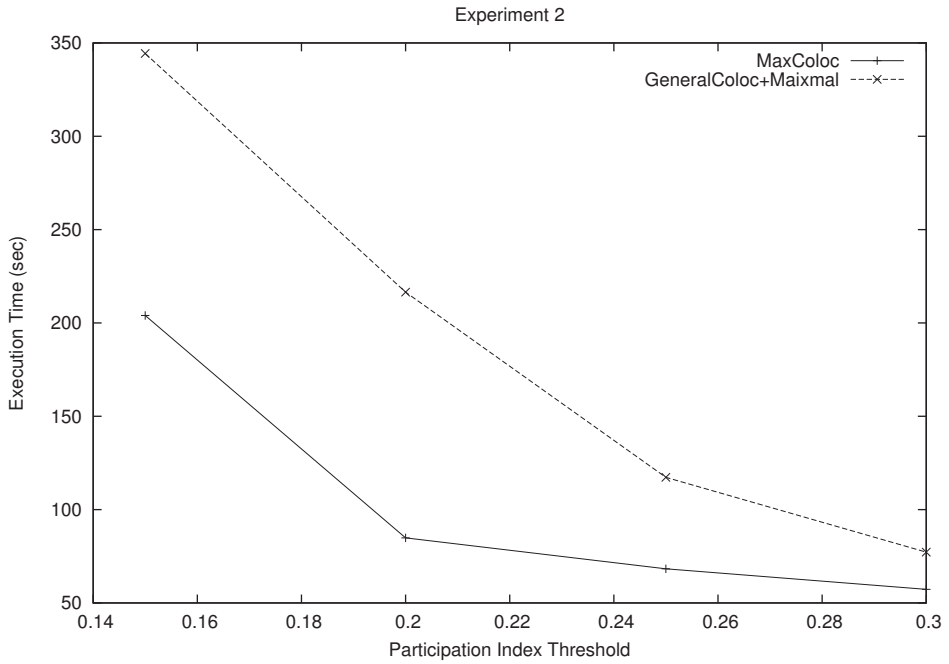


Figure 5.3. Experiment 2: By Prevalent Threshold

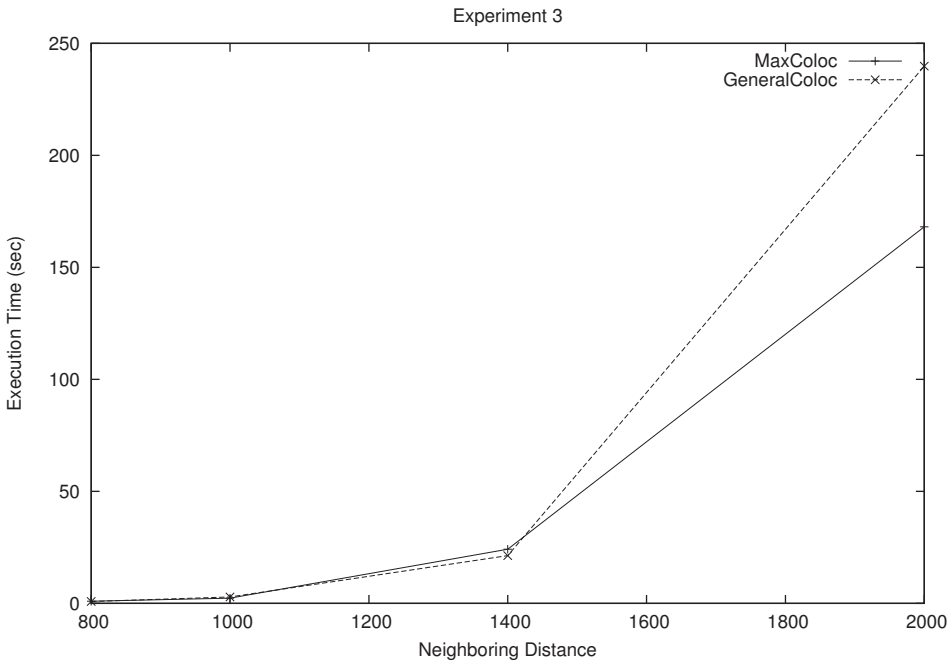


Figure 5.4. Experiment 3: By Neighboring Distance

that execution time of 'MaxColoc' is much less than that of 'GeneralColoc+Maximal'. When the prevalent threshold is 0.15, we see a large difference in execution time between the two algorithms. As the prevalent threshold increases, we see that difference in execution time of both algorithm decreases.

Experiment 3: When the neighboring distance increases in Figure 5.4, both algorithms' execution times increase. Initially, the execution time is about the same from a neighboring distance of 800 until a neighboring distance of 1200. At a neighboring distance of 1400, we see that the execution time 'GeneralColoc+Maximal' is slightly better than 'MaxColoc', but increases sharply when reaching a neighboring distance of 2000. The increase in neighboring distances makes the neighborhood area larger and increases the number of co-locations instances. The execution time of 'MaxColoc' also increases but not as quickly as 'GeneralColoc+Maximal'.

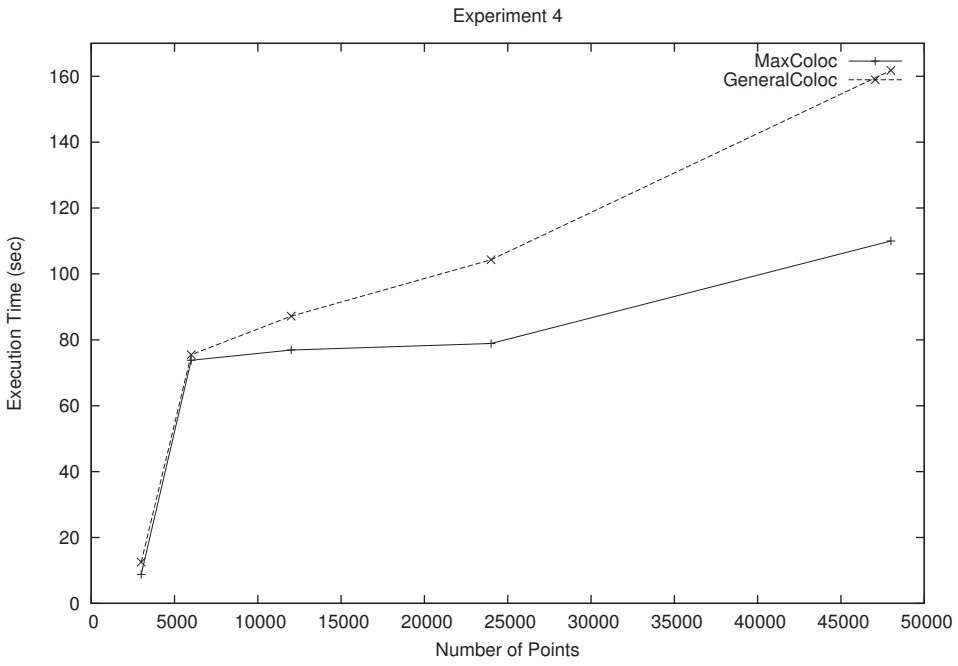


Figure 5.5. Experiment 4: By Number of Data Points

Experiment 4: When the number of data points increases in Figure 5.5, the execution time of both algorithms increases. The execution time of both algorithms increases quickly from 3000 to 6000 data points. From 6000 to 24000 data points, we note that the execution times increase slowly and then starting at 24000 to 48000 data points, the execution times sharply increase again. We see that execution time of 'MaxColoc' is less than that of 'GeneralColoc+Maximal' as the number of data points increases.

Algorithm 1 Maximal Co-location Algorithm

Inputs

$E = \{e_1, ..., e_n\}$: a set of spatial event types
S: a spatial dataset
d: a spatial neighbor relationship
θ: a minimum prevalence threshold

Variables

ST: a set of all neighborhood transactions.
$Tree_i$: CP-tree of type e_i
l: interest co-location size
C: a set of all candidate sets
C_l: a set of size l candidates
$upper_{pi}$: an approximate participation index of a candidate
pi: true participation index
l_{max}: the longest size of candidate c
CI_c: a set of clique instances of a candidates c
SI_c: a set of star instances of a candidate c
SI_l: a set of star instances of size l co-located event sets $SI_c \in SI_l$
R: a set of maximal co-located patterns
R_l: a set of size l maximal co-located patterns

Preprocess

1) ST=gen_neighbor_transactions(S, D);

Candidate generation

2) for $i = 1$ to m do
3) $Tree_i$ = build_event_tree(e_i, ST, θ);
4) end do
5) C = gen_candidates($Tree_1, ..., Tree_m$);
6) calculate_upper_pi(C);

Maximal co-location pattern mining

7) l_{max} = find_longest_size(C);
8) $l = l_{max}$
9) while($l \geq 2$ or $C_l \neq \emptyset$) do
10) C_l = get_l_candidates(C, l);
11) SI_l = find_star_instances(C_l, ST);
12) for each candidate c in C_l do
13) $upper_p i$ = Calculate_pi(SI_c);
14) if c.$upper_{pi} < \theta$ then continue;
15) CI_c = find_clique_instances(SI_l, c);
16) pi = calculate_pi(CI_c);
17) if $pi > \theta$
18) Insert(c, R_l);
19) end do
20) $R = R \cup R_l$; $C = C - C_l$;
21) Subset_Pruning(R_l, C);
22) $l = l - 1$;
23) end do

6 TOP- K CLOSED CO-LOCATION PATTERN MINING

This chapter shows the algorithmic design concepts of mining top-k closed co-location patterns and the algorithm. The chapter ends discussing the experimental results.

6.1 Background

A common framework of many spatial co-location mining algorithms required a user-defined a minimum prevalent threshold to discover interesting patterns. The problem is how to specify an appropriate user defined threshold. If the threshold is too high, the co-location mining algorithm might return too little result set or none at all. If the threshold is too low, it will generate too many patterns to the point where the task of analysing the patterns becomes impractical. To set an appropriate threshold would require the user to be knowledgeable of the data to be searched and the data mining task itself.

To overcome these problems, this thesis adopts the problem found in traditional data mining literature, of mining top-k closed co-location pattern [23]. Instead of requiring the user to specify a threshold, the user simply asks for the number of k closed co-location patterns to return. A co-location pattern is closed if it satisfies Definition 3.1.2 and Definition 3.1.3 for top-k closed.

6.2 Algorithmic Design Concept

If we take a brute force approach to mine top-k closed co-location patterns which involves finding all co-location instances forming clique neighboring relationships, compute the participation index per co-location, discover all closed co-locations by sort by their prevalence values and finally return top k closed result set, we would

have to examine around 2^l event sets when l event types are given. As the number of l event types increases, the brute force approach quickly becomes impractical. Our algorithmic framework focuses on reducing the number of co-locations examined for top-k co-location patterns.

6.2.1 Preprocess and Candidate Generation

The preprocess and candidate generation steps were explained in depth in Chapter 3. The spatial data set are converted to a set of neighboring transactions and event neighboring transactions. From the event neighborhood transactions, co-located candidates are generated.

6.2.2 Candidate Pruning

Top-k mining has additional pruning schemes to further reduce the co-location candidates. During the pattern mining process, an internal threshold θ_k is maintained to determine whether a candidate can be included in the result set. Even with the top-k closed result set is full with k closed co-location patterns, the result set must contain top-k co-location patterns. Using the internal threshold θ_k helps maintain top-k closed co-location pattern result set. θ_k maintains the smallest participation index in the result set. If the next co-location candidate's upper participation index is less than that of θ_k, the candidate is simply not included in the result set since the true participation index cannot be greater than the co-location candidate's upper participation index. The candidate is pruned as well as its supersets.

Figure 6.1 refers to the search spaces of top-k closed. In Figure 6.1 (a) show the generated co-location candidates along with their upper participation index where k is a user-defined parameter and the initial internal threshold, θ_k, is 0. Candidates are processed starting with length 2. Seven closed candidates are found as the grey shaded boxes indicate in Figure 6.1 (b). The internal threshold, θ_k, is set to 0.4 because it is the smallest participation index in the result set. The next candidate to be examined

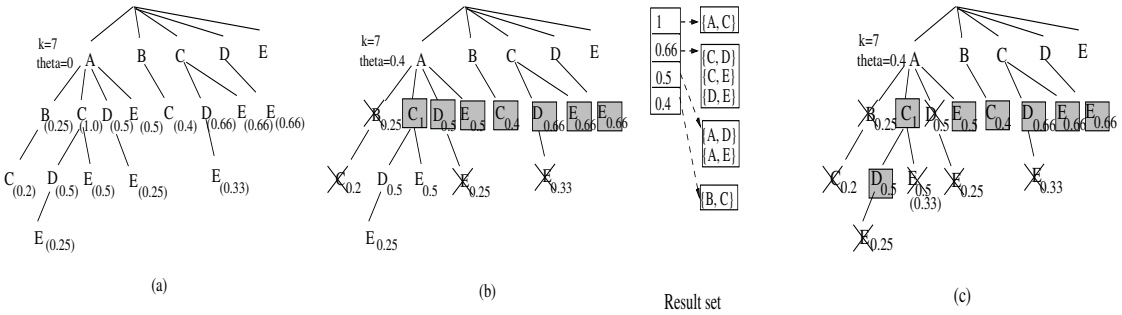


Figure 6.1. Top-k Closed Co-location Search Spaces

is {A, B} of Figure 6.1 (b). {A, B}, as well as its superset, is simply pruned as shown with the candidate marked out because the candidate's upper participation index is less than θ_k. {A, D, C} and {C, D, E} are also pruned out with the current θ_k value. By using this pruning scheme, the candidate search can be greatly reduced.

6.2.3 Co-location Instance Search

Once the co-location candidates are generated and filtered out any irreverent candidates, we must find the co-location instances of each candidate and compute their true participation index. Reuse the neighborhood transaction from the previous step, we can gather co-location instances from star instance of the neighborhood transaction and filter those instances having an additional relationship within the subset of that instance. The true participation index of a co-location is calculated from the participation ratios of their instances. This mining step has been discussed at length in chapter 3.

6.2.4 Closed Set Search

If a co-location candidate's true participation index is greater than the internal threshold θ_k, the candidate is included in the top-k closed result set. However, it has to examine whether the subsets in the result set whose participation index is the same according to definition 3.1.2 and definition 3.1.3. To efficiently search for subsets in the result set, the participation index is examined first. By check for the same participation index, we can avoid the operation of check for subset in the result set. The only time the subsets are checked is when the participation indexes are the same. Since the result set is sorted by participation index, we can employ a binary search for the first occurrence of the same participation index and then from there do a linear subset check until the participation index are not the same.

Going back to the example in Figure 6.1, (b) shows the current result set and search space with $k = 7$ and the internal $\theta_k = 0.4$. At this point length $l = 2$ co-location candidates have been examined and the search starts at the next level $l + 1$. Examining {A, C, D} with an upper participation index of 0.5, the algorithm determines that this candidate's participation index is greater than the θ_k and now must search the result set to see if there are any subsets that can be removed. By using definition 3.1.2, {A, B} with the participation index of 0.5 is removed and replace with the co-location {A, B, C}. The next candidate to examine is {A, C, E} with an upper participation index of 0.5 which passes our top-k pruning scheme but its calculated true participation index is 0.33. Comparing θ_k with the candidate's participation index, the candidate is pruned out. Examining candidates of length $l = 4$ shows that {A, C, D , E} is pruned out because of θ_k.

6.3 Algorithm

This section shows the pseudo-code algorithm to mine top-k closed co-locations based on the algorithmic design concepts. The pseudo-code is shown in Algorithm 2.

Preprocess (Step 1-2): Given an input spatial dataset and a neighbor relationship, first find all neighboring object pairs using a geometric method such as plane sweep [25], or a spatial query method using quaternary trees or R-trees [26]. The neighborhood transactions are generated by grouping the neighboring objects per each object.

Candidate generation (Steps 3-7): A candidate pattern tree (FP-tree) per each event type is constructed with the event neighborhood transactions. Star candidates are generated using a project based mining algorithm (FP-growth) [27]. Co-location candidates are filtered with combining the star candidates. The upper participation index of a candidate is computed with the support values of events in the set.

Preparing length l candidates (Step 8-13): The pattern mining process stats with length $l = 2$ event sets. An internal variable, minimum prevalence threshold θ_k, is set to 0 when $l = 2$. If the result set is full with k closed co-locations, we have a candidate pruning procedure before the mining processing of length $l+1$. If the upper participation index of candidate is less than θ_k, its superset as well as the candidate set are pruned out.

Gather the candidate instances (Step 14): The candidate instances of event sets are gathered from the neighborhood transactions.

Examine the terminate condition of length l processing (Step 15-17): If the result set is full during the length l processing, the upper participation indexes of the remaining candidates are compared with the internal threshold θ_k. When the upper participation index of current candidate set is less than θ_k, prune all remaining candidate sets of length l and their supersets and then go to step 32 for processing next candidates of length $l = l + 1$.

Filter the co-location instances of a candidate and the true participation index (Step 18-19): Otherwise, for each event set, filter its true co-location instances and calculate its true participation index.

Update the result set of top-k closed co-locations (Step 20-30): The update of the result set depends on whether the result set R is full or not. When the result set

R is not full, the candidate set can be included in the result set. If any subset of the current set is in the result set with the same participation index, the subset is replaced with the current set. Otherwise, the current event set is simply added in the result set. When the result set R is full and the participation index of the current candidate is greater than θ_k, the current set can also be inserted into the result set after checking its subsets having the same participation index. If the subset is found, it is replace with the current set. Otherwise, the last event set in the result set, the minimum prevalence threshold θ_k is always updated with the smallest participation index in the result set.

Return the final result set (Step 32-34): The procedure of step 9 to step 33 is repeated with an increase with pattern length, $l = l+1$. If there is no more candidates to process, return the final result set R.

6.4 Experimental Evaluation

This section shows the comparison of our top-k closed co-location algorithm with that of a general co-location mining algorithm.

6.4.1 Experimental Setup

We named our top-k closed co-location algorithm as 'TopKCColoc'. The general co-location algorithm we use is named 'GeneralColoc' which is found in [3]. 'GeneralColoc' has a user-defined minimum prevalence threshold parameter that filters out co-locations. That parameter is set to the smallest participation index of the result set of 'TopKCColoc'. 'GeneralColoc' also has a post processing step that generate top-k closed co-locations patterns. We named the general co-location pattern mining algorithm with the post processing module 'GeneralColoc+TopKClosed'. In the following experiments we compare the efficiency between 'TopKCColoc' and 'GeneralColoc+TopKClosed'.

We used the datasets containing points of interest in California [28] for our experiments. The number of original data points was 104,770 and the number of distinct event types was 40. We test 'TopKCColoc' and 'GeneralColoc+TopKClosed' for the following experiments:

- *Experiment 1:* Number of candidates: We observe the number of candidates examined for finding co-location instances of 'TopKCColoc' and 'GeneralColoc+ TopKClosed'. For this experiment, we prepared dataset #1 which has 40 event types, 12000 data points, and a neighboring distance of 1000. The values for k are 40 and 80. When $k = 40$ the smallest participation index when running 'TopKCColoc' is 0.17 and when $k = 80$, the smallest participation index is 0.09. These participation indexes were used as the minimum prevalent threshold for 'GeneralColoc+TopKClosed'.

- *Experiment 2:* Different k: We see the effects of runtime efficiency on different values of k for each execution of 'TopKCColoc and 'GeneralColoc+TopKClosed'. In our experiment, we prepared dataset #2 that contains 40 event types, 12000 points and a neighboring distance of 1000. By running this experiment, we observe the effect of runtime efficiency of 'TopKCColoc' and 'GeneralColoc+ TopKClosed' when k is set to 40, 60, and 80.

- *Experiment 3:* By neighboring distance: We see the effects of runtime efficiency as neighboring distance increases. For this experiment, we prepared datasets #3, #4, #5, and #6 each with a neighboring distance of 800, 100, 1400, 2000 respectively. These databases all have 40 event type and 1200 points. We test the experiment with $k = 40$ and $k = 80$.

- *Experiment 4:* By number of data points: We see the effects of runtime efficiency as the number of points increases. For this experiment, we prepared the dataset #7, #8, #9, #10 and #11 each having 3000, 6000, 12000, 24000, and 48000 respectively. Each experiment uses $k = 40$ and $k = 80$. All datasets have 40 event types and a neighboring distance of 1000.

All experiments were conducted on a Linux system with 2.0 GB main memory and the implementations of 'TopKCColoc' and 'GeneralColoc+TopKClosed' are in C++.

6.4.2 Experimental Results

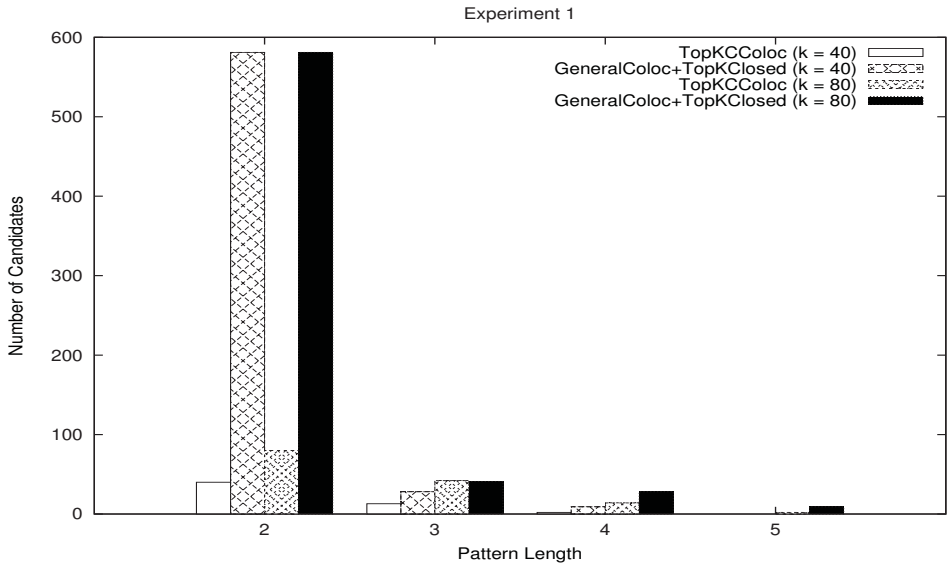


Figure 6.2. Experiment 1: Number of Candidates

Experiment 1: Figure 6.2 displays the number of candidates considered for co-location instances for each pattern length. Comparing the two algorithms show that the number of candidates examined for 'TopKCColoc' is much smaller compared to 'GeneralColoc+TopKClosed' both cases when k is set to 40 and 80. When k increases from 40 to 80, we see that 'GeneralColoc+TopKClosed' examines more candidates because of the minimum threshold decreases as shown in Figure 6.2 going from 0.17 to 0.09.

Experiment 2: As shown in Figure 6.3, comparing the execution time of both algorithms shows that the execution time of 'TopKCColoc' is less than that of 'GeneralColoc+TopKClosed'. The execution time of 'GeneralColoc+TopKClosed' increase

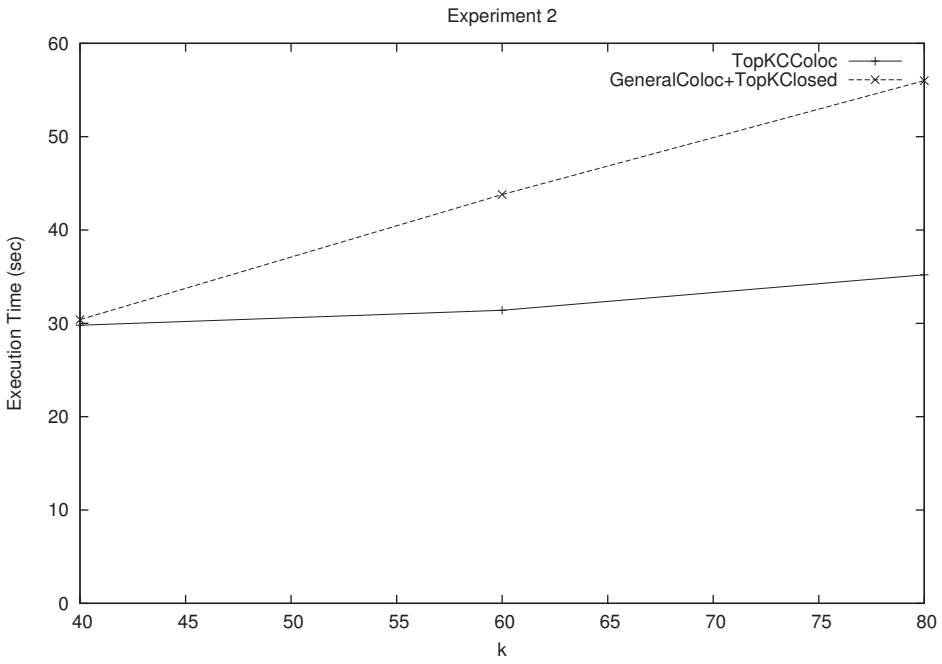


Figure 6.3. Experiment 2: By k

quickly as k increases. Likewise, the execution time of 'TopKCColoc' also increases as k increases but at a slower rate than that of 'GeneralColoc+TopKClosed'.

Experiment 3: In Figure 6.4, when $k = 40$, we see that from a neighboring distance of 800 to 1400 the execution times of 'TopKCColoc' and 'GeneralColoc+TopKClosed are about the same. From 1400, we see that 'GeneralColoc+TopKClosed' starts to increase slowly as k increases from a neighboring distance of 1400 to 2000 where as 'TopKCColoc' actually decreases on that same range. When $k = 80$, we see a sharp difference compared to the execution times when $k = 40$. The execution times of 'GeneralColoc+TopKClosed, where $k = 80$, increases very quickly starting from a neighboring distance of 1400 to 2000 in contrast to the execution times of 'TopKCColoc' which increases slowly.

Experiment 4: When $k = 40$, in Figure 6.5 the execution times of both 'TopKC-Coloc' and 'GeneralColoc+TopKClosed' slowly increases as the number of data points increases. 'GeneralColoc+TopKClosed' actually performs better than the 'TopKC-

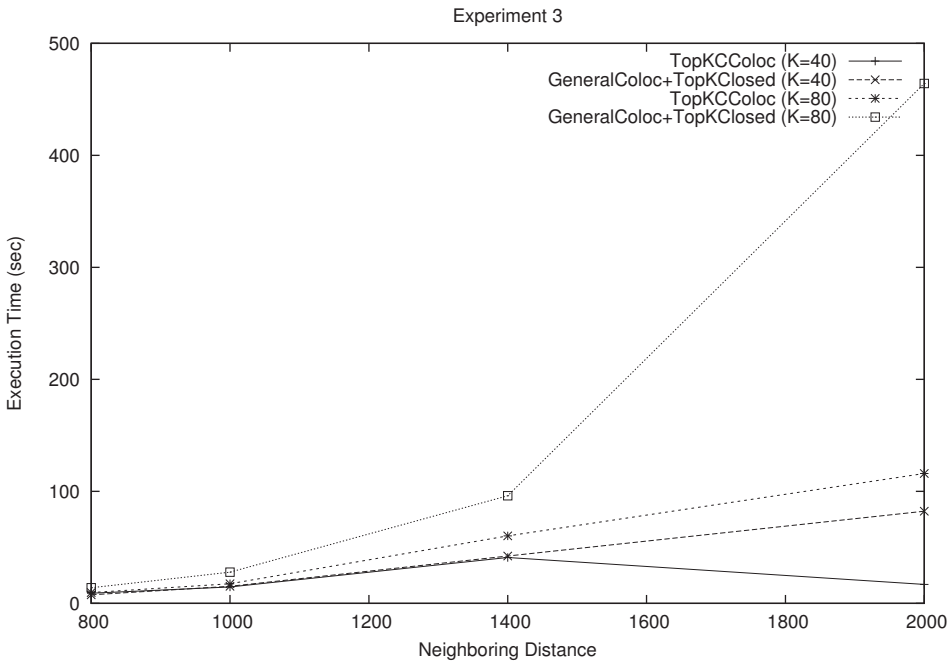


Figure 6.4. Experiment 3: By Neighboring Distance

Coloc'. But when $k = 80$, we see a huge difference in execution times between 'TopKCColoc' and 'GeneralColoc+TopKClosed'. The execution times of 'General-Coloc+TopKClosed' increase drastically as the number of data points increases where as the execution time of 'TopKCColoc' increases slowly and is much less than 'GeneralColoc+TopKClosed'.

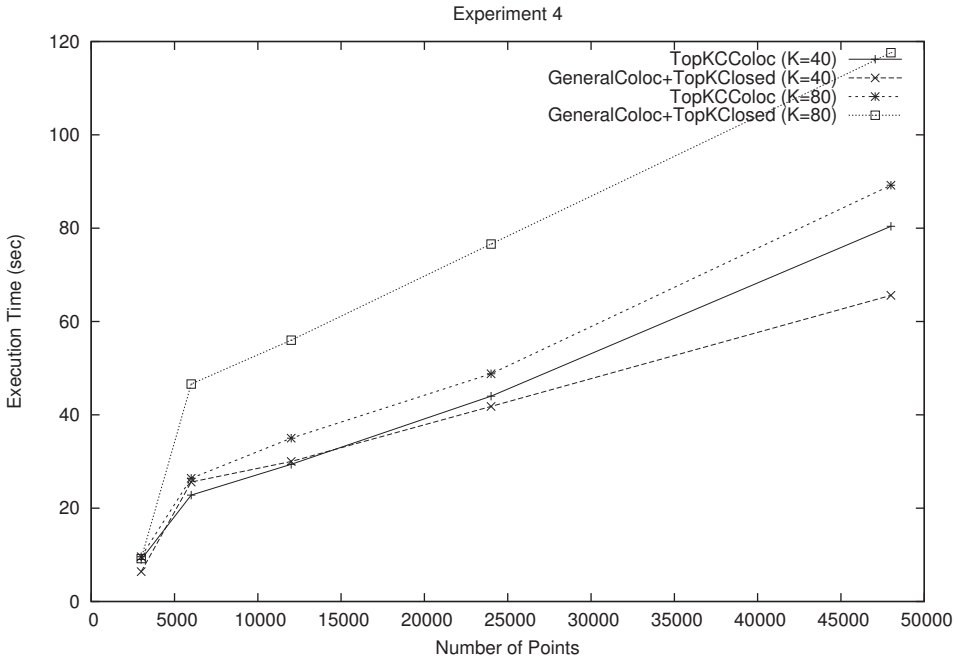


Figure 6.5. Experiment 4: By Number of Points

Algorithm 2 Top-k Closed Co-location Algorithm

Inputs

$E = \{e_1, ..., e_n\}$: a set of spatial even types

S: a spatial dataset

d: a spatial neighbor relationship

k: a number of closed co-locations of interest

Variables

ENT: a set of all neighborhood transactions

NP: a set of all neighbor parts (size 2)

$Tree_i$: the candidate pattern tree of type e_i

l: co-location length

θ_k: an internal minimum prevalence threshold for top-k closed co-located patterns

C: a set of all candidate sets

C_l: a set of length l candidate sets

$upper_{pi}$: an approximate participation index

pi: true participation index

CI_c: a set of clique instances of candidate c

SI_c: a set of start instances of candidate c

SI_l: a set of start instances of length l co-located event sets $SI_c \cup SI_l$

$subset$: a set of closed subsets of a candidate

R: a set of top-k closed co-located patterns, each set record has <event set, pi, rank>. The result set is sorted by the decreasing order of participation index

$R.last$: the last closed co-location in the result set

Preprocess and candidate generation

1) NP = find_neighbor_pairs(S, d);
2) (NT, ENT) = gen_neighbor_transactions(NP);
3) for i = 1 to m do
4) $Tree_i$ = build_candidate_pattern_tree(e_i, ENT);
5) end do
6) C = gen_candidates($Tree_i, ..., Tree_n$);
7) calculate_upper_pi(C);

top-k closed co-location mining

8) R = $\emptyset$; l = 2; θ_k = 0;
9) for no empty set C_l do
10) C_l = store_by_upper_pi(C_l, SI_i);
11) if $|R|$ == k then
12) prune_candidates(C, l, θ_k);
13) if C_l is $\emptyset$ then break;
14) SI_l = find_star_instances(C_l, NT);
15) for each candidate set $c \in C_i$ do
16) if $|R|$ == k and $c.upper\_pi < \theta_k$ then
17) prune_candidates(C, l, θ_k); continue;
18) CI_c = filter_clique_instances(SI_c);
19) pi = calculate_true_pi(CI_c);
20) if $|R| < k$ then
21) $subset$ = find_subset_closed(R, c, pi);
22) if $subset = \emptyset$ then insert(c, pi, R);
23) else replace(R, $subset$, c, pi);
24) else
25) if $\theta_k < pi$ then
26) $subsets$ = find_subset_closed(R, c, pi);
27) if $subsets = \emptyset$ then
28) replace(R, $R.last$, c, pi);
29) else replace(R, $subsets$, c, pi);
30) $\theta_k = R.last.pi$;
31) end do
32) $l = l + 1$;
33) end do
34) return R; exit;

7 CONCLUSION

This thesis proposed two problems for mining maximal and top-k closed co-located event sets. The mining result sets are much smaller than the result set mining of general co-located event sets because our result sets concisely represents co-location patterns. Algorithms to discover the compact co-location pattern were discovered. The algorithms ran several experiments for performance and the experiment results show that the proposed algorithms effectively mines maximal co-locations and top-k closed co-locations and perform more efficiently against general co-location mining algorithms with post processing modules. In the future, we plan to validate our algorithms with different types of datasets.

LIST OF REFERENCES

LIST OF REFERENCES

[1] J. F. Roddick and M. Spiliopoulou. A Bibliography of Temporal, Spatial and Saptio-Temporal Data Mining Research. In *Proc. of SIGKDD Explorations 1*, 1999.

[2] Miller H. J. and J. Han. *Geographic Data Mining and Knowledge Discovery*. Taylor and Francis, 2001.

[3] S. Shekhar and Y. Huang. Co-location Rules Mining: A Summary of Results. In *Proc. of International Symposium on Spatial and Temporal Database(SSTD)*, 2001.

[4] J. S. Yoo and S. Shekhar. A Partial Join Approach for Mining Co-location Patterns. In *Proc. of ACM ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems(ACM-GIS)*, 2004.

[5] J. S. Yoo and S. Shekhar. A Join-less Approach for Mining Spatial Co-location Patterns. *IEEE Transactions on Knowledge and Data Engineering*, 18(10), 2006.

[6] C. F. Eick, R. Parmar, W. Ding, T. F. Stepinski, and J. Nicot. Finding Regional Co-location Patterns for Sets of Continuous Variables in Spatial Datasets . In *Proc. of ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems(ACM-GIS)*, 2008.

[7] Y. Morimoto. Mining Frequent Neighboring Class Sets in Spatial Databases. In *Proc. of ACM SIGKDD Int'l Conf. on Knowledge Discovery and Data Mining*, 2001.

[8] K. Koperski and J. Han. Discovery of Spatial Association Rules in Geographic Information Databases. In *Proc. of International Symposium on Large Spatial Data bases, 47-66*, 1995.

[9] W. Ding, R. Jiamthapthaksin1, R. Parmar, D. Jiang, T. F. Stepinski, and C. F. Eick. Towards Region Discovery in Spatial Datasets . In *Proc. of Pacific-Asia Conference on Knowledge Discovery and Data Mining (PAKDD)*, 2008.

[10] X. Xiao, X. Xie, Q. Luo, and W. Ma. Density Based Co-location Pattern Discovery. In *Proc. of ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems(ACM-GIS)*, 2008.

[11] S. Shekhar, P. Zhang, Y. Huang, and R. Vatsavai. *Trends in Spatial Data Mining, as a book chapter in Data Mining: Next Generation Challenges and Future Directions*. AAAI/MIT Press, 2004.

[12] V. Castro and A. Murray. Discovering Assoications in Spatial Data-An Efficient Medoid Based Approach. In *Conference on Knowledge Discovery and Data Mining (PAKDD)*, 1998.

[13] J.S. Yoo and S. Shekhar. A Join-less Apporach for Spatial Co-location Mining: A Summary of Results. In *Proc. of Fifth IEEE International Conference on Data Mining(ICDM)*, 2005.

[14] X. Zhang, N. Mamoulis, D. Cheung, and Y. Shou. Fast Mining of Spatial Collocations. In *Proc. of ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2004.

[15] J. S. Yoo and M. Bow. Finding N-Most Prevalent Colocated Event Sets: A Summary of Results . In *Technical Report, http://users.ipfw.edu/yooj/publications/NMost-Summary.pdf*, 2009.

[16] J. S. Yoo and M. Bow. Mining Spatial Colocation Patterns Without Thresholds. In *Technical Report, http://users.ipfw.edu/yooj/publications/NMost-Summary.pdf*, 2009.

[17] Roberto J. Bayardo Jr. Efficiently Mining Long Patterns From Databases. In *Proc. of the 1998 ACM-SIGMOD Int'l Conf. on Management of Data*, 1998.

[18] M. J. Zaki, S. Partharsarathy, M.Ogihara, and W. Li. New Algorithms for Fast Discovery of Association Rules. In *Proc. of the 1998 ACM-SIGMOD Int'l Conf. on Management of Data*, 1998.

[19] Dao-I Lin and Zvi M.Kedem. Pincer-Search: A new Algorithm for Discovering the Maximum Frequent Set. In *In Proc. of the internalational Conference on Extending Database Technology*, 1997.

[20] D. Burdick, M. Calimlim, and J. Gehrke. MAFIA: A Maximal Freqent Itemset Algorithm for Transactional Databases. In *Proc. of the 1998 ACM-SIGMOD Int'l Conf. on Management of Data*, 1998.

[21] Andrea Pietracaprina and Fabio Vandin. Efficient Incremental Mining of Top-K Frequent Closed Itemsets. In *Discovery Science: 10th International Conference Proceedings*, 2007.

[22] Panida Songram and Veera Boonjing. Mining Top-K Closed Itemsets Using Best-First Search. In *Proceedings of 2008 IEEE 8th International Conference on Computer and Information Technology*, 2008.

[23] J. Wang, J. Han, Y. Lu, and P. Tzvetkov. TFP: An Efficient Algorithm for Mining Top-K Frequent Closed Itemsets. *IEEE Transactions on Knowledge and Data Engineering*, 17(5), 2005.

[24] Y. Hirate, E. Iwahashi, and H. Yamana. TF^2P-growth: An Efficient Algorithm for Mining Frequent Patterns without any Thresholds . In *Proc. of Workshop on Alternative Techniques for Data Mining and Knowledge Discovery*, 2004.

[25] M Berg, M. Kreveld, Overmars. M, and O. Schwarzkopf. *Computational Geometry*. Springer, 2000.

[26] S. Shekhar and S. Chawla. *Spatial Databases: A Tour*. Prentice Hall, 2003.

[27] J. Han, J. Pei, and Y. Yin. Mining Frequent Patterns Without Candidate Generation. In *Proc. of ACM SIGMOD Conference on Management of Data*, 2000.

[28] F. Li, D. Cheng, M. Hadjieleftheriou, G. Kollios, and S. Teng. On Trip Planning Queries in Spatial Databases . In *Proc. of Interational Symposium on Advances in Spatial and Temporal Databases(SSTD)*, 2005.